案例 5-圆中点 Bresenham 算法

文档编写：霍波魏

校稿/修订：孔令德

时间 2019~2020

联系方式：QQ997796978

说明：本套案例由孔令德开发，原版本为 Visual C++6.0，配套于孔令德的著作《计算机图形学-基于 MFC 三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于 Windows 10 操作系统，使用 Microsoft visual studio 2017 平台的 MFC（英文版）开发。

一、知识点

1. CCircle 类

设计画圆的 CCircle 类，用八分法画圆。

2. 八分画圆法

Bresenham 画圆算法又称中点画圆算法，与 Bresenham 直线算法一样，其基本的方法是利用误差项来判断选择最近的像素点。为了简便，考虑一个圆心在坐标原点的圆，而且只计算八分圆周上的点，其余圆周上的点利用对称性就可得到。

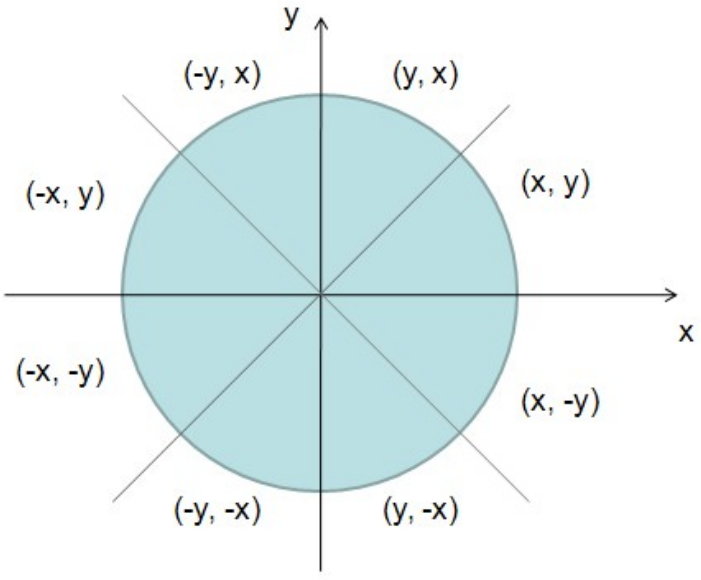

图 5-1 圆的“八对称性”

显然，只需要知道了圆上的一个点的坐标 (x, y) ，利用对称性，我们马上就能得到另外七个对称点的坐标。

二、案例描述

本案例通过确定圆心，绘制 30 个以坐标原点为圆心、半径相差 20 的同心圆，并通过半径的递增，使得同心圆由小变大，并形成一个循环。

三、实现步骤

1. 在 CTestView 类中添加 DoubleBuffer()双缓冲函数。
2. 添加 CCircle 类并在其中用八分法画圆。
3. 在 CTestView 类中添加 DrawObject 函数，并在其中进行同心圆绘制。
4. 在 CTestView 类的构造函数中对变量进行初始化。
5. 在 OnDraw()中对 SetTimer()和 DoubleBuffer()进行调用。

四、主要算法

1.CCircle 类

```
public:
    CCircle();
    ~CCircle();
    void SetData(int r, CP2i p0);
    void Draw(CDC* pDC);
    void CirclePoint(CP2i p, CDC* pDC);
public:
    int r;//半径
    CRGB clr;//颜色
    CP2i CenterPoint;//圆心
void CCircle::DrawCircle(CDC * pDC, CRGB clr)//圆中点 Bresenham 算法
{
    CP2i p;
    double d;
    d = 1.25 - r; p.x = 0, p.y = r;
    for (; p.x <= p.y; p.x++)
    {
        CirclePoint(p, pDC, clr);//调用八分法画圆子函数
        if (d < 0.0)
            d += 2 * p.x + 3;
        else
        {
            d += 2 * (p.x - p.y) + 5;
            p.y--;
        }
    }
}
```

```
void CCircle::CirclePoint(CP2i p, CDC * pDC, CRGB clr)//八分法画圆子函数
{
    COLORREF c = CRGBTORGB(clr);
    pDC->SetPixelV(p.x + CenterPoint.x, p.y + CenterPoint.y, c);//x,y
    pDC->SetPixelV(p.y + CenterPoint.x, p.x + CenterPoint.y, c);//y,x
    pDC->SetPixelV(p.y + CenterPoint.x, -p.x + CenterPoint.y, c);    //y,-x
    pDC->SetPixelV(p.x + CenterPoint.x, -p.y + CenterPoint.y, c);    //x,-y
    pDC->SetPixelV(-p.x + CenterPoint.x, -p.y + CenterPoint.y, c);  //-x,-y
    pDC->SetPixelV(-p.y + CenterPoint.x, -p.x + CenterPoint.y, c);  //-y,-x
    pDC->SetPixelV(-p.y + CenterPoint.x, p.x + CenterPoint.y, c);   //-y,x
    pDC->SetPixelV(-p.x + CenterPoint.x, p.y + CenterPoint.y, c);   //-x,y
    }
```

2.CTestView 类

```
public:
    void DoubleBuffer(CDC* pDC);//双缓冲
    void DrawObject(CDC* pDC);//绘制图形
protected:
    BOOL bPlay;//动画开关
    CCircle circle[30];
    int nRadius;//圆的半径
void CTestView::DoubleBuffer(CDC* pDC)//双缓冲
{
    CRect rect;//定义客户区矩形
    GetClientRect(&rect);//获得客户区的大小
    pDC->SetMapMode(MM_ANISOTROPIC);//pDC 自定义坐标系
    pDC->SetWindowExt(rect.Width(), rect.Height());//设置窗口范围
    pDC->SetViewportExt(rect.Width(), -rect.Height());
                    //设置视区范围,x 轴水平向右，y 轴垂直向上
    pDC->SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
                    //客户区中心为原点
    CDC memDC;//内存 DC
    CBitmap NewBitmap, *pOldBitmap;//内存中承载的临时位图
    memDC.CreateCompatibleDC(pDC);//创建一个与显示 pDC 兼容的内存 memDC
    NewBitmap.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
                    //创建兼容位图
    pOldBitmap = memDC.SelectObject(&NewBitmap);//将兼容位图选入 memDC
    memDC.SetMapMode(MM_ANISOTROPIC);//memDC 自定义坐标系
    memDC.SetWindowExt(rect.Width(), rect.Height());
    memDC.SetViewportExt(rect.Width(), -rect.Height());
    memDC.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);
    rect.OffsetRect(-rect.Width() / 2, -rect.Height() / 2);
    DrawObject(&memDC);//向 memDC 绘制图形
    pDC->BitBlt(rect.left, rect.top, rect.Width(), rect.Height(), &memDC,
        -rect.Width() / 2, -rect.Height() / 2, SRCCOPY);
```

```
                    //将内存 memDC 中的位图拷贝到显示 pDC 中
    memDC.SelectObject(pOldBitmap);//恢复位图
    NewBitmap.DeleteObject();//删除位图
}
void CTestView::DrawObject(CDC* pDC)//绘制
{
    CRGB clr = CRGB(1.0, 1.0, 0.0);//设置背景色
    for (int i = 0; i < 30; i++)
    {
        circle[i].SetData(nRadius + i * 20, CP2i(0, 0));
        circle[i].DrawCircle(pDC, clr);
    }
}
```

五、实现效果

圆中点 Bresenham 算法效果如图 4-2 所示。

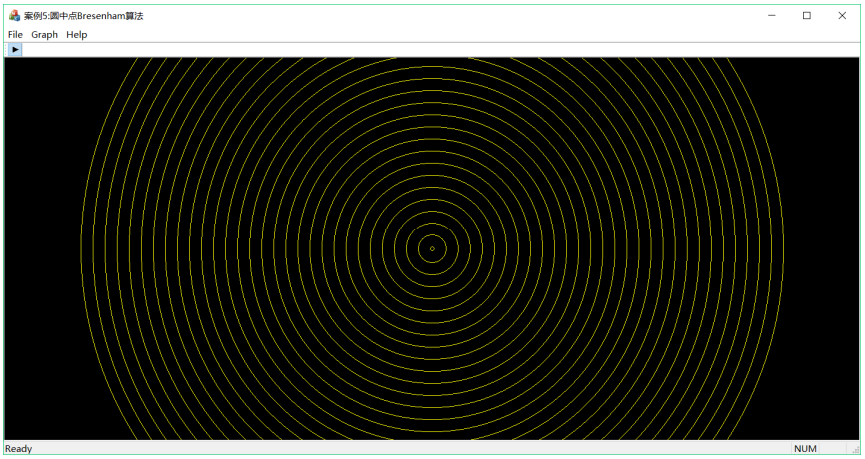

图 5-2 圆中点 Bresenham 算法效果图

六、案例心得

通过本案例更加熟悉了 Bresenham 算法，不仅仅是在直线上时候用 Bresenham 算法，在其他各种图形中都可以使用；并了解了八分画圆算法。